AN49943

PSoC® 1 USB-to-UART Bridge

Author: Barry Gackle
Associated Project: Yes
Associated Part Family: CY8C24x94
Software Version: PSoC® Designer™ 5.2
Related Application Notes: None

Abstract

Although USB is now the generally accepted standard for interfacing with PCs, RS-232 and other UART protocols are still widely used in embedded systems and some PC software. This application note describes a USB-to-UART bridge solution implemented in PSoC® 1. User module configuration, critical firmware, and reasons for a USB-to-UART bridge are also described.

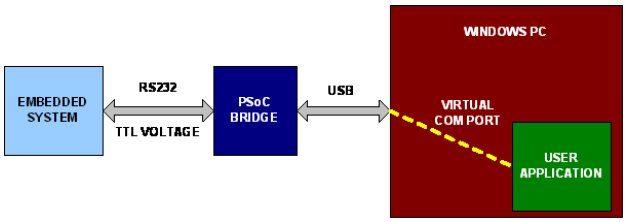
Introduction

This application note shows how to implement a full featured USB-to-UART bridge using PSoC 1. This solution is designed as a reference for designers who need such a bridge in their system. This solution can be implemented and tested on a CY3214 USB Evaluation Board.

Description

The USB-to-UART bridge sits between an embedded system and a host PC. The embedded system connects to the bridge through a standard UART interface. The PC connects to the bridge through the USB. From the perspective of the embedded system, the bridge presents a PC UART port. From the perspective of the application software on the host PC, the bridge also enumerates as a UART. With this arrangement, you can reuse the existing firmware and application software designed to communicate through the UART. This arrangement also allows embedded software engineers who are familiar with programming for a UART, to develop USB solutions without having to learn additional programming techniques. The following figure shows a high-level block diagram of a system that incorporates the USB-to-UART bridge.

Figure 1. USB-to-UART System



The bridge supports the following baud rates: 2400, 4800, 9600, 19200, 38400, 57600, and 115200 bps. You can also implement custom baud rates by modifying the code in the included example project. The bridge also supports dynamic baud rate changes from the PC terminal software, while the device is operating.

The data format is one start bit, 8 data bits, an optional parity bit, and one stop bit. If the parity bit is used, it can be even or odd. Software (XON/XOFF) and hardware (CTS/RTS) flow control are supported. The burst data rate limit for UART transmission from the PSoC 1 device is 6 Mbps, although sustained speeds may be slower because of data processing time. The data rate on the USB side of the bridge is determined by the host.

The bridge itself is designed to be powered by the USB port. With minimal hardware and software changes, the bridge can be redesigned to be self-powered. These changes are discussed later in this application note.

Purpose

Building a USB-to-UART bridge into a system is necessary in several situations. Any existing system that uses a UART to communicate with a PC can have a bridge installed to permit communication over USB with no further firmware modification. Because the bridge enumerates on the PC side as a UART, any existing support software can be used with no or minimal modifications.

Communication through USB is important for connected embedded systems. Most consumer PCs sold today do not contain RS-232 ports. Microsoft considers hardware RS-232 support to be deprecated in the Windows operating system. To retain compatibility with modern PC hardware, systems must replace RS-232 communication

with USB. The USB-to-UART bridge solution allows replacing RS-232 communication with USB with maximum reuse of existing systems.

Using PSoC 1 in this role allows flexibility and reduces the BOM cost, both of which are not possible with single-purpose bridge chips. The design described in this application note keeps a significant portion of the PSoC 1 resources free for other system uses. The solution, together with this application note, is specifically designed to be easy to extend, which minimizes the NRE involved with integrating additional features.

BOM reduction is also possible because of differences in voltage shifting requirements. RS-232 generally requires voltage level shifting to interface with the standard 3 V and 5 V logic systems. With PSoC 1, the UART interfaces directly with 3 V or 5 V TTL systems, and also directly with the USB. Adding PSoC 1 to an embedded system that currently uses RS-232 eliminates voltage-shifting hardware in most cases. Figure 2 shows a block diagram of a system that uses voltage level shifting for RS-232 communications. Figure 3 shows a similar system that uses a bridge to eliminate the voltage-shifting requirement.

Figure 2. RS-232 System with Level Shifting

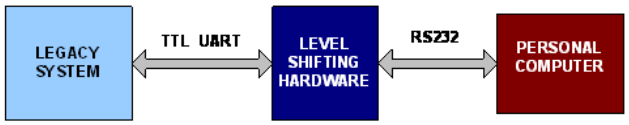
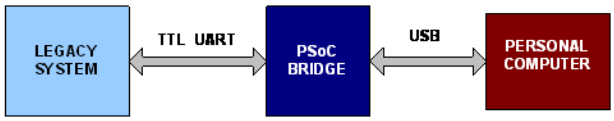


Figure 3. Level Shifting Replaced by USB Bridge



Although this solution enables the use of PSoC 1 as a bridge with no additional firmware development, there is considerable flexibility with a PSoC 1–based solution that is not possible with a single-purpose bridge chip. Because the bridge functionality is written entirely in firmware, it is possible to make arbitrary changes in functionality.

Architecture

Device Setup: Global Resources Settings

Figure 4 shows the PSoC Global Resources settings. The key settings to note are power, system clock, VC1, and the watchdog setting.

In this particular example, power is shown as 5 V, which is appropriate for a bus-powered system. Changing this to 3.3 V is also possible, but requires slight changes in the code used to initialize the USBUART user module.

VC1 ultimately drives the UART baud clock. If changes are made to the VC1 divider, then the counter values in the source code must change accordingly to maintain the desired baud rate.

The Analog Power setting is set to SC On/Ref Low. This is because the analog blocks are used to detect USB bus power.

Finally, the watchdog is enabled. Code is provided to reset the watchdog periodically.

All other settings work as shown, but can be easily changed to accommodate additional system requirements with minimal impact on the functionality described in this application note.

Figure 4. Global Resource Settings

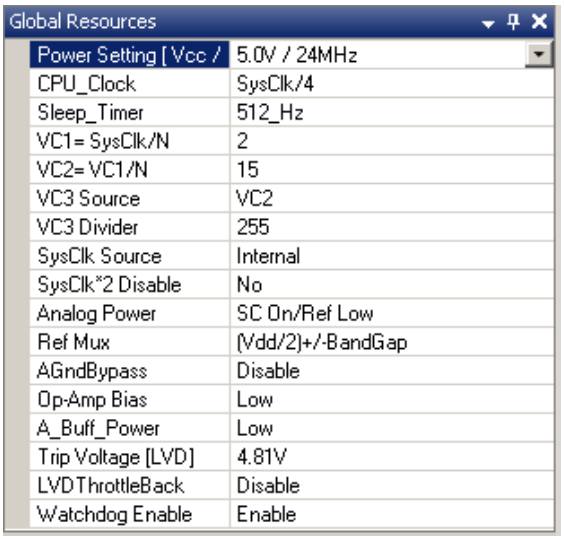
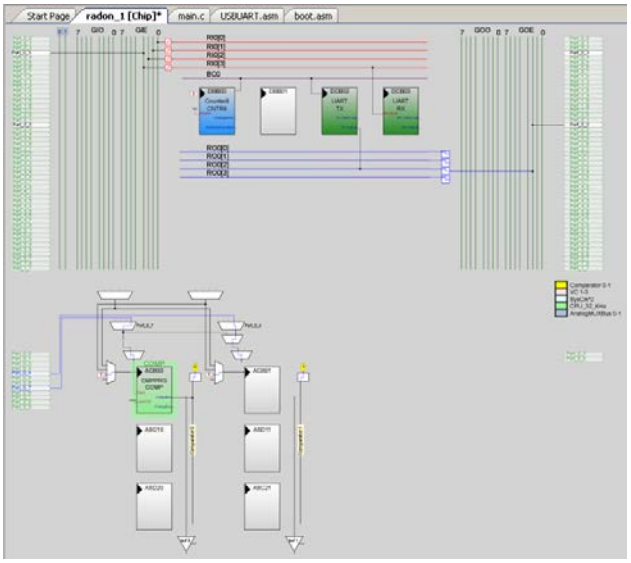


Figure 5. User Module Layout



Device Setup: User Modules

Figure 5 shows the user module layout for this project. This design uses four PSoC 1 user modules: an 8-bit counter, a UART, a USBUART, and a programmable comparator. The 8-bit counter serves as the clock for the UART. The UART baud clock can be set to a specific value by changing the counter value. The UART is used as the embedded (RS-232) side of the bridge. The USBUART is used as the PC (USB) side of the bridge. The programmable comparator is used to detect the presence of bus power from the USB.

Figure 6. Counter8 User Module Properties

Properties - Counter8

Name	Counter8
User Module	Counter8
Version	2.5
Clock	VC1
ClockSync	Sync to SysClk
Enable	High
CompareOut	None
TerminalCountOut	None
Period	155
CompareValue	78
CompareType	Less Than Or Equal
InterruptType	Terminal Count
InvertEnable	Normal

Figure 6 shows the user module properties for the 8-bit counter. The counter is clocked by VC1. The counter period is set to produce the desired UART baud clock rate. Note that the UART baud clock must be eight times the desired data rate. The compare value must be set to a value that is roughly half the period, to produce a square wave with a 50% duty cycle. The counter enable bit is connected to V_{CC}, so that the counter is always on.

Although it is not apparent from the Module Properties window, the compare output of the counter is connected to Row Broadcast Bus 0. This connection is set by clicking BC0 and setting it to DBB00. Because the counter Terminal Count output and interrupt are not used in this application, the remaining properties are not important. You can change them to meet a specific system requirement.

Figure 7 shows the user module properties for the UART. The key settings are the RX input, TX output, Clock, RX Command Buffer, and the TX interrupt mode. The RX input and TX output can be changed if you need different pins for the UART side of the bridge. The RX Command Buffer must be disabled. This application works in part by making changes to the user module code driving the UART RX, and is not compatible with the RX Command buffer, an optional higher-level UART API. The TX Interrupt Mode must be set to TXRegEmpty for proper operation. The InvertRX Input option is set to normal in this example. This can be enabled if appropriate for the system with which it communicates. The remaining settings must have no impact on bridge operation.

Figure 7. UART User Module Properties

Parameters - UART

Name	UART
User Module	UART
Version	5.3
Clock	Row_0_Broadcast
RX Input	Row_0_Input_2
TX Output	Row_0_Output_2
TX Interrupt Mode	TXRegEmpty
ClockSync	Sync to SysClk
RxCmdBuffer	Disable
RxBufferSize	16
CommandTerminator	13
Param_Delimiter	0
IgnoreCharsBelow	0
Enable_BackSpace	Disable
RX Output	None
RX Clock Out	None
TX Clock Out	None
InvertRX Input	Normal

Figure 8 shows the user module properties for the programmable comparator. The comparator receives a voltage from the VBUS line of the USB connector. This voltage is used to determine whether the bus is active. The USB specification requires that a device does not send power back to an inactive USB bus. This mechanism detects whether the bus is powered and prevents the PSoC 1 device from placing voltage on the bus if it is. This is done as part of the USB Compliance Checklist to ensure that the device's pull up is active only when VBUS is high.

Figure 8. CMPPRG User Module Properties

Properties - CMPPRG

Name	CMPPRG
User Module	CMPPRG
Version	3.3
AnalogBus	Disable
CompBus	ComparatorBus_0
Input	AnalogColumnMUXBusSwitch_0
LowLimit	VSS
RefValue	0.750

Figure 9 shows the user module properties for the USBUART user module. This module does not show up on the user module layout portion of the Chip View because the USB hardware is dedicated. As a result, no

specific placement and routing are required. This user module handles all communications with a PC through USB. It enumerates as a serial communication device and shows up to the host computer as a virtual UART port. VendorID and ProductID must be set to appropriate values for the required end use. If you do not have a VendorID assigned to your company, see the following site for information on obtaining one from the USB Implementers Forum: http://www.usb.org/developers/vendor

Note that the USB Implementers Forum is a separate entity not affiliated with Cypress Semiconductor. Vendor string values are purely descriptive strings and must be set to appropriate values describing the system.

Serial numbering is an optional feature. If you change the system from bus-powered to self-powered, you must also change the DevicePower setting. Finally, if you use the system as bus powered and add additional functionalities, the maximum power setting may need to be increased.

Figure 9. USBUART User Module Properties

Properties - USBUART

Name	USBUART
User Module	USBUART
Version	1.2
VendorID	0000
ProductID	0000
VendorString	Cypress Demo
ProductString	Cypress USB to UART Demo
SerialNumberType	None
SerialNumberString	0000
DevicePower	Bus Powered
MaxPower	100

Software

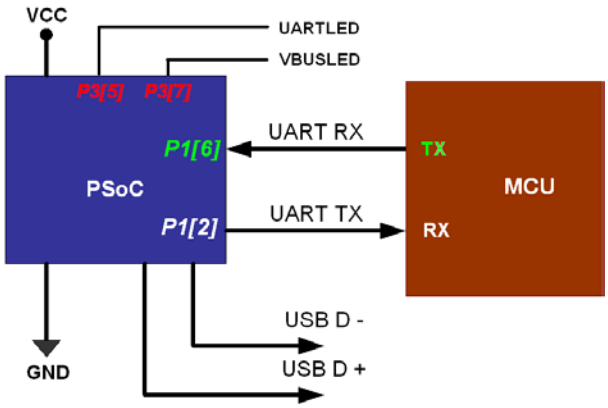
The bridge software consists primarily of two concurrent loops. The background loop is implemented in the UART RX and TX interrupts and handles reception and transmission of data over the UART. The foreground loop is an infinite while() loop located in *main.c*. This loop handles USB transmission and reception, and initiates UART transmission when USB data is present. The foreground loop also handles dynamic data rate switching and USB bus power detection.

Any user customization of the software must be added to the foreground loop. You must ensure that the execution length of this loop is not longer than the time to transmit one character over the UART. Failing to do so can result in UART buffer overflow. Awareness of timing becomes critical if any additional interrupt routines are added, because failure to service UART interrupts in a timely manner also results in buffer overflows.

The UART firmware is altered to include dual receive buffers for incoming UART data. This allows one buffer to be copied out to the USB port while a second receives data.

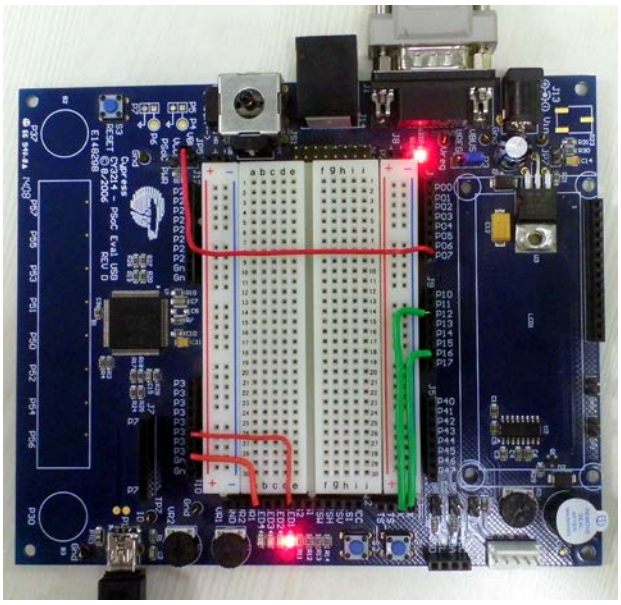
Testing

Figure 10. Testing the USB-UART Bridge Using CY3214



The example USB-to-UART project can be tested with the CY3214 kit and any MCU with a UART interface. Figure 10 shows the block diagram for a typical USB-UART bridge application. In order to quickly test the project with just the CY3214 kit and a PC, the UART side of the bridge can be looped back to the PC using an RS232 interface. The USB end of the bridge is connected to the PC through a USB cable and the UART end is connected to an RS232 port on the kit. Figure 11 shows a snapshot of the system setup using the CY3214 kit. Pins P12 and P16 correspond to the TX and RX pins of the bridge. These are connected to the PC serial port using the RS232 connector (J1). The board is connected to the PC USB port using connector P1. The kit is powered over USB.

Figure 11. Snapshot of USB-UART Setup on CY3214 Kit



The following connections are to be made:

1. Connect one end of an RS232 serial cable to J1 of CY3214 kit and other end to serial port of PC
2. Connect P16 of J9 to RX header of J3
3. Connect P12 of J9 to TX header of J3
4. Connect P35 of J10 to LED1 of J2
5. Connect P37 of J10 to LED4 of J2
6. Connect VBUS header of JP5 to P07 of J4
7. Place jumper on P3 to select VBUS as power source
8. Connect one end of a USB mini-B cable to the PC and the other end to USB connector on the kit (P1) to power the kit from USB port

When the project is loaded to the board and powered using the USB cable, the PC will recognize it as “Cypress USB to UART demo”. In order to enumerate the project as a USB to UART, the OS should be directed to the Setup Information (*.INF*) file generated by PSoC designer. To do this, manually direct the OS to the “lib” folder of the AN49943 project. The project will now correctly enumerate as a virtual COM port.

To test the bridge functionality, open two instances of a terminal application like HyperTerminal or TeraTerm. Configure both the instances with identical serial settings. When any character is typed on one terminal, it will be detected by the bridge and will be displayed on the other terminal instance.

There are two status LEDs that can be used for debugging purposes.

LED1 (VBUSLED) connected to P35 will glow whenever power on the USB line is detected. VBUS power is monitored by P07, which is routed to the internal comparator.

LED4 (UARTLED) glows initially during startup, till the USBUART user module initializes successfully. During normal operation, LED4 should flash once whenever the UART configuration is changed for the enumerated COM port.

Summary

The USB-to-UART bridge serves as an effective bridge between an embedded system and a host PC, when used with PSoC 1. This bridge helps systems to retain compatibility with modern PC hardware. Using PSoC 1 to implement this provides flexibility and is also cost effective.

Document History

Document Title: PSoC® 1 USB-to-UART Bridge – AN49943

Document Number: 001-49943

Revision	ECN	Submission Date	Orig. of Change	Description of Change
**	2653927	01/04/2009	GYV/AESA	New application note
*A	2873532	02/04/2010	RLRM	Updated project files
*B	3153453	01/25/2011	KLMZ	Updated Software Version on page 1 as PSoC Designer™ 5.1 SP1 Changed title to “PSoC® 1 USB to RS-232 Bridge” Updated abstract
*C	3164413	02/07/2011	KLMZ	Added associated files
*D	3202226	03/21/2011	KLMZ	Updated to describe a general USB-to-UART bridge instead of a bridge specific to RS-232
*E	3478728	12/29/2011	KLMZ	Updated template Updated attached project
*F	3565595	04/12/2012	ARVI	Updated project to support dynamic baud rate changes, VBUS detection

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com